岡山理科大学紀要 第41号 A pp. 59-68(2005)

戦略パラメータを使用しない効率的EPについて

成久　洋之・太田　真由美・谷口　隆裕*・片山　謙吾

岡山理科大学工学部情報工学科

*岡山理科大学大学院工学研究科情報工学専攻

(2005年9月30日受付、2005年11月7日受理)

要約

進化的計算法 [1] の一つである EP(Evolutionary Programming) は関数最適化問題に対して有効な手法であると考えられている。この EP は解の探索幅と探索方向を規定するために戦略パラメータを使用している。本論文は戦略パラメータを使用しない指数型 EP(nsEEP) を提案し、その有効性を検討したものである。数値実験の結果、この新手法はかなり効率的なものであることが判明した。

1　はじめに

進化的プログラミング (Evolutionary Programming;EP)[4] は遺伝的アルゴリズム (Genetic Algorithm;GA) などと同様に生物の進化の過程と自然の選択を模倣した最適化手法で、複雑で大規模な問題に対する解法としては極めて有効なものと考えられている。GA も EP も個体集団 (population) を形成し、集団で精度の良い最適近似解を求めようとする集団最適化手法であるが、GA の主要演算 (オペレータ) が交叉 (crossover) であるのに対し、EP のそれは突然変異 (mutation) のみ使用するものである。したがって、実数を主体とした乱数を使用した突然変異となるので機械的に処理しやすい利点が指摘されている。

EP は L.J.Fogel が人工知能に対するアプローチとして提案したもので、後に D.B.Fogel によって最適化手法として開発されたものである。この EP は Gauss Mutation (Gauss 分布に従う乱数を使用した突然変異により解を進化させるもの) を主体にしたものであり、これを CEP と呼んでいる。[2][5] 1965 年に、Yao et al. は Cauchy Mutation (Cauchy 分布に従う乱数を使用した突然変異により解を進化させるもの) を主としたものを提案し、これを FEP と呼んでいる。これら CEP と FEP とでは全般的に FEP の方が有効な EP であるとみなされている。

2002 年に、 Narihisa et al. は Exponential Mutation (複合指数分布に従う乱数を使用した突然変異により解を進化させるもの) を主体としたものを提案し、これを EEP (Exponential Evolutionary Programming) と呼んでいる。[7][8][9][10] これは指数分布のパラメータを進化の状態に対応させて、初期段階では広域探索に、最適解の近傍に近づけば局所探索向けに制御しようとすることで進化の効率化を狙った EP 手法である。これまでの研究では分布パラメータのとり方として、固定値をとる Fix EEP と分布パラメータを多段階に変動させる Multi-Switching EEP を提案し、CEP や FEP に比較して有効な EP であることを示した。さらに、分布パラメータを連続的に変動させる linEEP(パラメータを線形に変動させる) や expEEP(パラメータを指数関数的に変動させる) を提案し、それらの有効性を示してきた。

本論文は、従来の EP 手法で重視されている戦略パラメータを使用しない expEEP(これを nsEEP と呼ぶ) を新規に提案し、その有効性につき検討したものである。

2　指数型進化的プログラミング (EEP)

EEP は分布パラメータ λ に従う複合指数乱数 $E(0,\lambda)$ を使用した突然変異 (exponential mutation) により解を進化させるアルゴリズムであり、その概要は次のとおり。

Step1 (初期個体集団の生成)： μ 個の n 次元実数ベクトル対 $(\boldsymbol{x}_i, \boldsymbol{\sigma}_i)$, $i = 1, 2, ..., \mu$ を生成する.

Step2 (各個体の適応度評価)： $f(\boldsymbol{x}_i)$, $i = 1, 2, ..., \mu$ を計算する.

Step3 (子孫の生成)：各個体 $(\boldsymbol{x}_i, \boldsymbol{\sigma}_i)$ から単一の子孫 $(\boldsymbol{x'}_i, \boldsymbol{\sigma'}_i)$ を次のように生成する．

$$\sigma'_i(j) = \sigma_i(j) exp[\tau' N(0,1) + \tau N_j(0,1)], \tag{1}$$

$$x'_i(j) = x_i(j) + \sigma'_i(j) E_j(0,\lambda), \tag{2}$$

$$i = 1,2,...,\mu, \quad j = 1,2,...,n$$

ただし、$x_i(j)$, $x'_i(j)$, $\sigma_i(j)$, $\sigma'_i(j)$ はそれぞれ、ベクトル $\boldsymbol{x}_i$, $\boldsymbol{x'}_i$, $\boldsymbol{\sigma}_i$, $\boldsymbol{\sigma'}_i$ の j 番目の成分を示す．$N(0,1)$ は正規乱数 (平均=0, 標準偏差=1) を表し、$N_j(0,1)$ は各 j 毎に新たに乱数を発生させるものである．$\tau = (\sqrt{2\sqrt{n}})^{-1}$, $\tau' = (\sqrt{2n})^{-1}$ とする．$E(0,\lambda)$ はパラメータ λ の複合指数乱数 (平均=0, 分散= $\frac{2}{\lambda^2}$) を示し、$E_j(0,\lambda)$ は j 毎に新規に乱数を発生させることを意味する．

Step4 (子孫の適応度評価)：子孫 $(\boldsymbol{x'}_i, \boldsymbol{\sigma'}_i)$ における $f(\boldsymbol{x'}_i)$, $i = 1,2,...,\mu$ を計算する．

Step5 (各個体のトーナメント評価)： 2μ 個の親と子孫 $\{ (\boldsymbol{x}_i, \boldsymbol{\sigma}_i), (\boldsymbol{x'}_i, \boldsymbol{\sigma'}_i) \}$, $i = 1,2,...,\mu$ の中から任意の q 個をランダムに選ぶ．これらの q 個の $\boldsymbol{x}$ ベクトルを $\boldsymbol{s}_1, \boldsymbol{s}_2, ..., \boldsymbol{s}_q$ とする．$f(\boldsymbol{x}_i) \geq f(\boldsymbol{s}_k)$ であれば個体 $(\boldsymbol{x}_i, \boldsymbol{\sigma}_i)$ は勝ち点 1 点を取得する．すべての $i(i = 1,2,...,2\mu)$ について、$k = 1,2,...,q$ の比較を行う．

Step6 (次世代の親の選択)：Step5 での勝ち点の多い方から μ 個選択し次世代の親とする．

Step7 (停止・続行判定)：終了条件を満たせば処理停止, そうでなければ Step2 へ戻る．

3 EEP における λ の決定法

EEP は複合指数乱数による突然変異を使用するので、指数分布におけるパラメータ λ の値がその収束特性に多大の影響を与える。これまでの研究では次のようなものを検討してきた。

(1) 固定値： λ の値を全進化世代で一定値に固定する。(Fix EEP)

(2) 多段階変動値：全進化世代で λ を複数回変動させ、世代が進むにつれ λ の値を増大させる。(Multi-Switching EEP)

(3) 増加関数値： λ の値を世代 g の増加関数として与えるもので、これには線形関数 λ_L と指数関数 λ_E とがある。

$$\lambda_L = \lambda_1 + \frac{\lambda_2 - \lambda_1}{G} g \tag{3}$$

$$\lambda_E = \lambda_1 exp[(ln\frac{\lambda_2}{\lambda_1})\frac{g}{G}] \tag{4}$$

ただし、 λ_1 は λ の初期値、λ_2 は λ の最終値、G は全進化世代をそれぞれ示すものとする。λ として λ_L を使用するものを linEEP、 λ_E を使用するものを expEEP と呼んでいる。λ の値として増加関数を考えることで λ_L および λ_E を使用した場合、 λ のパラメータ値として λ_1 と λ_2 のみを設定すればよく、EEP におけるパラメータ数を減少させ得る利点がある。このことは、最適化アルゴリズムという観点から極めて望ましいことである。

4 expEEP における λ の役割

EEP アルゴリズムにおける Step3 での (1)、(2) から

$$x'_i(j) = x_i(j) + \sigma_i(j) E_j(0,\lambda) exp[\tau' N(0,1) + \tau N_j(0,1)] \tag{5}$$

として子孫 $x'_i(j)$ が生成される。ここで $E_j(0,\lambda)$ として、(4) の λ_E を使用すると

$$x'_i(j) = x_i(j) + \sigma_i(j) E_j(0,1) S,$$

$$S = \frac{1}{\lambda_1} exp[(-ln(\frac{\lambda_2}{\lambda_1}))\frac{g}{G} + \tau' N(0,1) + \tau N_j(0,1)], \tag{6}$$

として表せる。本来 (5) における

$$\sigma_i(j)E_j(0,\lambda)exp[\tau' N(0,1)+\tau N_j(0,1)]$$

は自己適応機能 (self-adaptivity) と呼ばれているもので、進化の状況に応じて減少しているものである。したがって、(6) の S における $(-ln(\frac{\lambda_2}{\lambda_1}))\frac{g}{G}$ が $\tau' N(0,1)+\tau N_j(0,1)$ の値に連動したものでなければ、EP における自己適応機能を弱めることになる。このことから、EP におけるこれらの機能を補強するような λ_1, λ_2 でなければ、EEP の効率化は期待できないことになる。戦略パラメータとしての $\sigma_i(j)$ は進化が進むにつれて小さくなる特性を持ち、これは指数的減少傾向を示している。実際の数値計算では局所解に陥ることを避けるため、その下限 ϵ を設定し、$\sigma < \epsilon$ であれば $\sigma = \epsilon$ として計算させている。一方、$\frac{1}{\lambda_E} = \frac{1}{\lambda_1}exp[(-ln(\frac{\lambda_2}{\lambda_1}))\frac{g}{G}]$ も指数関数的に減少し、G においては $\frac{1}{\lambda_2}$ となる。そこで、(6) 式における自己適応機能を満足させるような、λ_1, λ_2 を決定することは実験的に求めるにしてもかなり労力を必要とすることから、$\sigma_i(j)$ を抜きにした λ_1, λ_2 の適当値を探す方が労力は少なくて済むのではないかとも考えられる。

5 戦略パラメータなしの expEEP(nsEEP)

前節における考え方に基づき、分布パラメータ λ として、$\lambda = \lambda_E$ とした戦略パラメータを使用しない expEEP を nsEEP と呼び、次のような nsEEP アルゴリズムを提案する。

Step1 μ 個の初期個体集団 $\{\boldsymbol{x}_i\}$, $i = 1, 2, ..., \mu$, $\boldsymbol{x}_i \in \boldsymbol{R}^n$ を生成する.

Step2 $f(\boldsymbol{x}_i)$, $i = 1, 2, ..., \mu$ を計算する.

Step3 各個体 $\boldsymbol{x}_i$ から単一の子孫 $\boldsymbol{x'}_i$ を次のように生成する.

$$\begin{aligned} x'_i(j) &= x_i(j) + E_j(0,\lambda) \\ E_j(0,\lambda) &= \frac{1}{\lambda}E_j(0,1) \\ \lambda &= \lambda_1 exp[(ln(\frac{\lambda_2}{\lambda_1}))\frac{g}{G}] \\ i &= 1, 2, ..., \mu, \quad j = 1, 2, ..., n \end{aligned} \tag{7}$$

Step4 $f(\boldsymbol{x'}_i)$, $i = 1, 2, ..., \mu$ を計算する.

Step5 q トーナメント選択により、$\{\cup \boldsymbol{x}_i\} \cup \{\cup \boldsymbol{x'}_i\}$ から μ 個の個体を選び次世代の親とする.

Step6 終了条件を満たせば処理停止, そうでなければ Step2 へ戻る.

6 数値実験

6.1 対象問題

本研究での対象問題は、この分野の最適化問題としてよく知られているベンチマーク問題とし、表1で与えた12問題とする。

表1 適用問題

Test Function	S	f_{min}
$f_1(x) = \sum_{i=1}^{30} x_i^2$	$[-100, 100]^{30}$	0
$f_2(x) = \sum_{i=1}^{30} \|x_i\| + \prod_{i=1}^{30} \|x_i\|$	$[-10, 10]^{30}$	0
$f_3(x) = \sum_{i=1}^{30} (\sum_{j=1}^{i} x_j)^2$	$[-100, 100]^{30}$	0
$f_4(x) = max\{\|x_i\|, 1 \leq i \leq 30\}$	$[-100, 100]$	0
$f_5(x) = \sum_{i=1}^{29} [100(x_{x+1} - x_i^2)^2 + (x_i - 1)^2]$	$[-30, 30]^{30}$	0
$f_6(x) = \sum_{i=1}^{30} (\lfloor x_i + 0.5 \rfloor)^2$	$[-100, 100]^{30}$	0
$f_7(x) = \sum_{i=1}^{30} i x_i^4 + random[0, 1)$	$[-1.28, 1.28]^{30}$	0
$f_8(x) = -\sum_{i=1}^{30} (x_i sin(\sqrt{\|x_i\|}))$	$[-500, 500]^{30}$	-12569.5
$f_9(x) = \sum_{i=1}^{30} [x_i^2 - 10cos(2\pi x_i) + 10]$	$[-5.12, 5.12]^{30}$	0
$f_{10}(x) = -20exp[-2.0\sqrt{\frac{1}{30}\sum_{i=1}^{30} x_i^2}] - exp[\frac{1}{30}\sum_{i=1}^{30} cos2\pi x_i] + 20 + e$	$[-32, 32]^{30}$	0
$f_{11}(x) = \frac{1}{4000}\sum_{i=1}^{30} x_i^2 - \prod_{i=1}^{30} cos(\frac{x_i}{\sqrt{i}}) + 1$	$[-600, 600]^{30}$	0
$f_{12}(x) = \frac{\pi}{n}\{10sin^2(\pi y_i) + \sum_{i=1}^{n-1}(y_i - 1)^2[1 + 10sin^2(\pi y_{i+1})] + (y_n - 1)^2\}$ $+ \sum_{i=1}^{n} u(x_i, 10, 100, 4), \quad y_i = 1 + \frac{1}{4}(x_i + 1)$	$[-50, 50]^{30}$	0

6.2 実験要領

(1) 個体集団数： $\mu = 100$

(2) トーナメントサイズ： $q = 10$

(3) 進化世代数： $G = 5000$

(4) 適応度：100 runs の平均

(5) 分布パラメータ λ ： $\lambda_1 = 5.0 \times 10^{-2}$, $\lambda_2 = 5.0 \times 10^{10}$

(6) 性能比較のために使用する CEP、FEP のパラメータ
$\epsilon = \sigma$ の下限 $= 10^{-4}$
なお、 σ の上限は 1.0 とする.

(7) 実験環境
Intel Pentium4 CPU 2.53GHz RAM 256MB

7 実験結果とその考察

本論文で提案した戦略パラメータを使用しない EEP としての nsEEP を関数最小化問題に適応した結果を表 2 に示す。

表 2 最終世代での関数値

	CEP	FEP	nsEEP
f_1	0.000127883	0.117415256	**4.76138E-20**
f_2	0.037526577	0.982577878	**9.52647E-10**
f_3	2.698873419	15.32974318	**0.06214049**
f_4	15.82811054	0.087222395	**9.35564E-11**
f_5	86.37205982	96.95708171	**42.08864914**
f_6	170.5788	0.3533	**0**
f_7	0.060257634	0.122782395	**0.016950557**
f_8	-7326.263085	**-12437.68989**	-8908.172791
f_9	**11.30438623**	27.20929974	48.29853828
f_{10}	1.361045781	0.364342106	**1.59959E-10**
f_{11}	0.177531732	0.018376979	**0.005468791**
f_{12}	1.28601905	0.0008712	**3.14684E-22**

表 2 は、nsEEP の収束解を示すが、参考のために従来の EP 手法としての CEP および FEP を適用した場合の収束解をも表示した。この結果、12 問中 10 問において nsEEP が最良解を得ており、CEP と FEP がそれぞれ 1 個の最良解となっている。特に、nsEEP は f_1 から f_6 での単峰性関数においてはすべての問題に対して最良解を得ている。しかも、f_1, f_2, f_4, f_{10} および f_{12} に対しては CEP や FEP に比較にならないくらいの精度の良い解となっていることがわかる。

表 3 は、5000 世代までの処理時間を示す。ただし、単位は秒とする。

表 3 処理時間の比較

	CEP	FEP	nsEEP
f_1	1091	1015	195
f_2	1108	1033	207
f_3	1370	1296	417
f_4	1118	1033	206
f_5	1103	1030	208
f_6	1216	1141	281
f_7	1092	1024	196
f_8	1340	1273	367
f_9	1282	1193	334
f_{10}	1298	1206	343
f_{11}	1323	1275	366
f_{12}	1370	1286	416

nsEEP は個体として戦略パラメータを含まないので各世代の進化において、そのための進化処理が不必要となる。このことから、少なくとも $\frac{1}{2}$ 以上の処理時間の短縮は期待しうるものであったが、表 3 に示すように CEP や FEP に比較して $\frac{1}{3}$ から $\frac{1}{6}$ の処理時間となっている。

以上の実験結果より、nsEEP は分布パラメータ (λ_1, λ_2) について、$\lambda_1 = 5.0 \times 10^{-2}$, $\lambda_2 = 5.0 \times 10^{10}$ とセットするだけで、従来の CEP や FEP に比較して極めて効率的な EP アルゴリズムであるといえる。

なお、図1から図12において各関数の収束特性を示す。図1は f_1 に対する収束特性を示すものである。nsEEPはほとんど全世代を通じてほぼ同じ収束率を示している。800世代まではFEPの方が良いが、最終的にはFEPは最悪となっている。これは如実にCEPとFEPの収束特性を表しているものといえる。すなわち、Cauchy MutationはGaussian Mutationに対して探索率が大きいがために進化の初期段階では威力を発揮するが、終期の段階では、親と子孫との距離が親と最適解との距離を超過している結果といえる。図4は f_4 の収束特性を示すが、nsEEP, FEP, CEPの順によい収束特性となっている。図5は f_5 の収束特性を示すが、1000世代以降においてnsEEPが他を凌駕している。図6は f_6 の収束特性を示すが、1038世代で最適解に収束している。図7は f_7 の収束特性を示すが、nsEEPにおいて200〜1000世代でstagnationが見られるものの、それ以降は良好な収束を実施している。図8、図9はそれぞれ f_8, f_9 の収束特性を示すが、図8ではFEPがbestで、図9ではCEPがbestな収束を表している。図10、図12においては、nsEEPは全世代を通じて一貫した収束状況を示している。図11では、nsEEPの収束は1500世代以降において飽和状況を示しているが、最終解はnsEEPが良い結果を表している。f_3 に対しては2000世代、f_5 に対しては1500世代、f_7 に関しては2500世代、f_8 に対しては500世代、f_9 に対しては1500世代以降において、nsEEPの収束は飽和状況を示しており、分布パラメータ λ の値が大きすぎていることを示している。すなわち、それらの各世代で局所解に陥っているものと考えられる。

参考文献

[1] T.Back and H.-P.Schwefel, "An overview of evolutionary algorithms for parameter optimization", Evolutionary Computation, 1(1):pp.1-23, 1993.

[2] X.Yao, Y.Liu, "Fast Evolutionary Programming", Proc. of the 5th Annual Conference on Evolutionary Programming, MIT Press, pp.451-460, 1996.

[3] K.-H.Liang, X.Yao, Y.Liu, C.Newton and D.Hoffman, "An Experimental Investigation of Selfadaptation in Evolutionary Programming", Proc. of the 7th Annual Conference on Evolutionary Programming, Springer-Verla, pp.291-300, 1998.

[4] K.Chellapilla, "Combining Mutation Operators in Evolutionary Programming", IEEE Transactions on Evolutionary Computation, 2(3), pp.91-96, 1998.

[5] X.Yao, Y.Liu, G.Lin, "Evolutionary Programming Made Faster", IEEE Transactions on Evolutionary Computation, 3(2), pp.82-102, 1999.

[6] C.-Y.Lee and Y.song, "Evolutionary Programming using the Levy probability Distribution", Proc. of Genetic and Evolutionary Computation Conference (GECCO'99), Morgan Daufman, pp.886-893, 1999.

[7] K.Kohmoto, H.Narihisa, K.Katayama, "Evolutionary Programming Using Exponential Mutation", Proc. of the 6th World Multiconference on Systematics, Cybernetics and Informatics, vol.11, Computer Science 2, July 14-18, USA, pp.405-410, 2002.

[8] H.Narihisa, K.Kohmoto and K.Katayama, "Evolutionary Programming with Double Exponential Probability Distribution", Proc. of the Second International Association of Science and Technology for Development (IASTED) International Conference on Artificial Intelligence and Applications (AIA2002), pp.358-363, 2002.

[9] K.Kohmoto, H.Narihisa and K.Katayama, "PERFORMANCE OF EVOLUTIONARY PROGRAMMING USING EXPONENTIAL MUTATION", Proc. of The 4th Asia-Pacific Conference on Simulated Evolution And Learning (SEAL2002), vol.2, pp.454-458, 2002.

[10] H.Narihisa, K.Kohmoto, T.Kumon and K.Katayama, "Performance of Exponential Evolutionary Programming", Proc. of the 7th IASTED International Conference on Artificial Intelligence and Soft Computing (ASC2003), pp.243-248, 2003.

[11] H.Narihisa, K.Kohmoto, M.Tsuda, K.Katayama, "CONVERGENCE CHARACTERISTICS OF EXPONENTIAL EVOLUTIONARY PROGRAMMING", Proc. of the 8th IASTED International Conference on Artificial Intelligence and Soft Computing (ASC2004), pp.426-431, 2004.

[12] H.Narihisa, T.Taniguchi, M.Thuda and K.Katayama, "Efficiency of Parallel Exponential Evolutionary Programming", Proc. of the 2005 International conference on Parallel Processing Workshops, pp.585-595, 2005.

[13] H.Narihisa, T.Taniguchi, M.Ohta and K.Katayama, "EVOLUTIONARY PROGRAMMING WITH EXPONENTIAL MUTATION", Proc. of the Ninth IASTED International Conference on ARTIFICIAL INTELLIGENCE AND SOFT COMPUTING (ASC2005), pp.55-60, 2005.

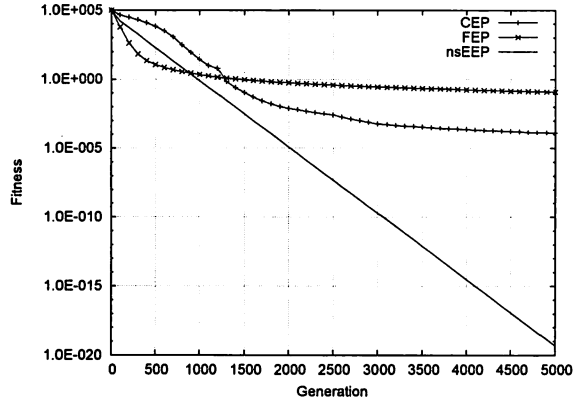

図 1　f_1 の収束特性

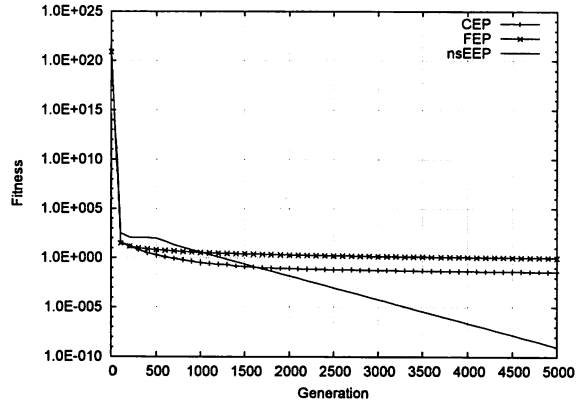

図 2　f_2 の収束特性

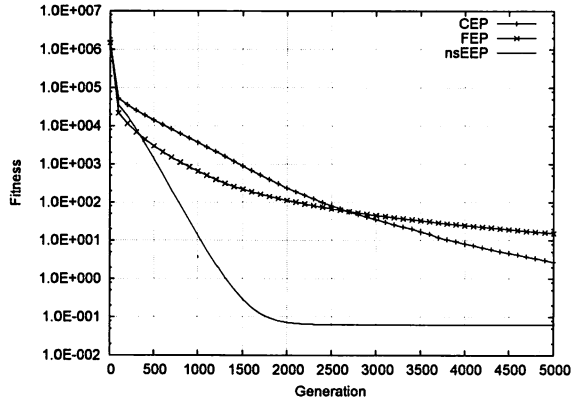

図 3　f_3 の収束特性

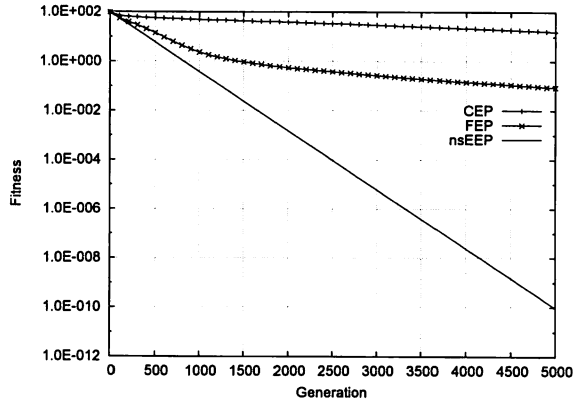

図 4　f_4 の収束特性

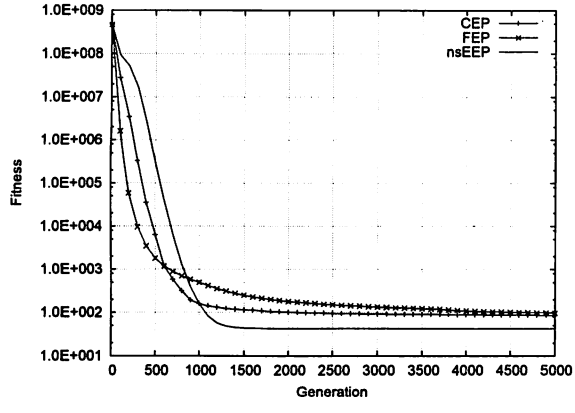

図 5　f_5 の収束特性

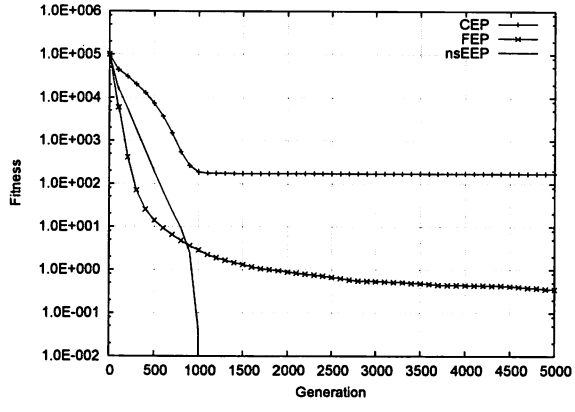

図 6　f_6 の収束特性

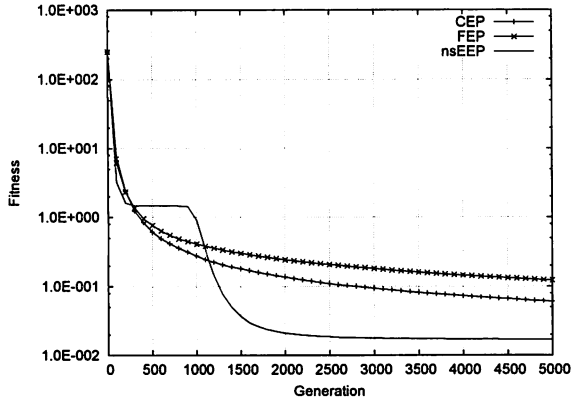

図 7　f_7 の収束特性

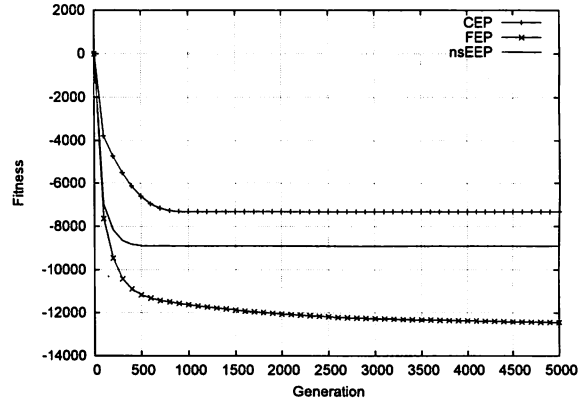

図 8　f_8 の収束特性

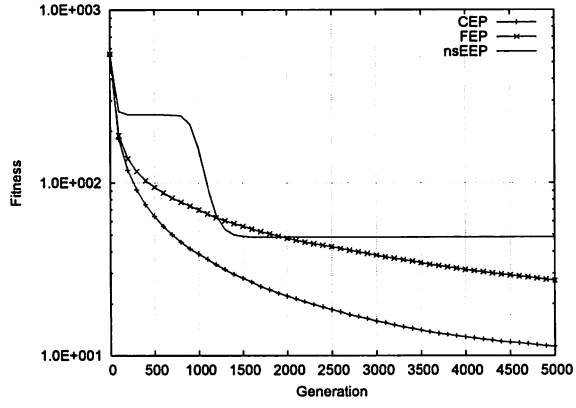

図 9　f_9 の収束特性

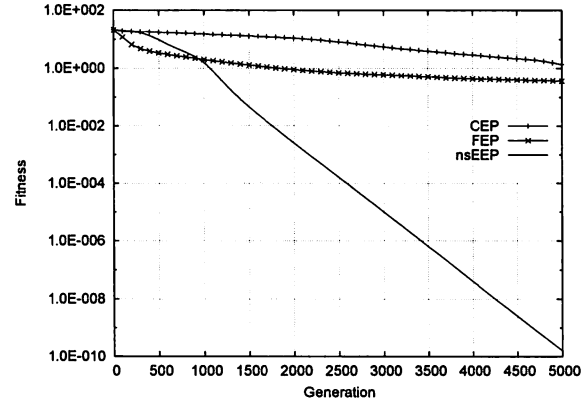

図 10　f_{10} の収束特性

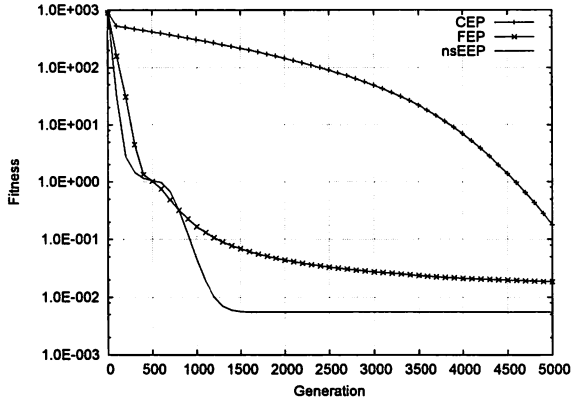

図 11　f_{11} の収束特性

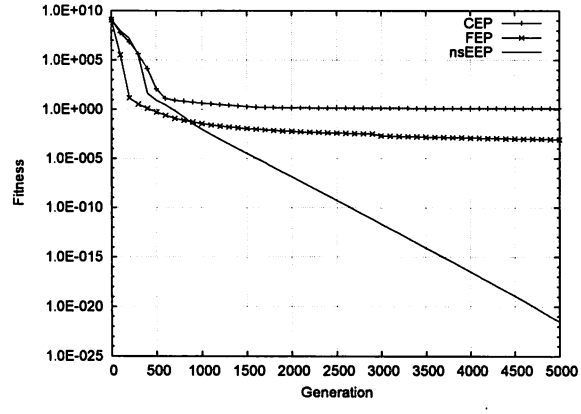

図 12　f_{12} の収束特性

An Efficient Evolutionary Programming Without Using Strategy Parameter

Hiroyuki Narihisa, Mayumi Ohta,
Hirotaka Taniguchi* and Kengo Katayama

Department of Information and Computer Engineering,
Faculty of Engineering,
**Graduate School of Engineering,*
Okayama University of Science,
1-1 Ridai-cho, Okayama, 700-0005, Japan

(Received September 30, 2005; accepted November 7, 2005)

Evolutionary Programming which uses only mutation operator is considered to be an efficient approach to solve function optimization problems. Conventionally, Evolutionary Programming uses strategy parameter for self-adaptation of function optimization.

In this paper, we present an efficient new EP algorithm without using strategy parameter by adopting exponential distribution parameter of EEP. Essentially, EEP(Exponential EP) uses exponential mutation. EEP needs to determine appropriate distribution parameter value. Our newly proposed EP(nsEEP) provides a similar role as conventional strategy parameter by using appropriate exponential distribution parameter value. The results of numerical experiments show that nsEEP outperforms conventional EP algorithm such as CEP and FEP.